Enhanced Cuckoo Search Algorithm for Virtual Machine Placement in Cloud Data Centers

Esha Barlaskar[1], Yumnam Jayanta Singh[2] and Biju Issac[3]
[1]School of Electronics, Electrical Engineering and Computer Science,
Queen's University Belfast, UK
[2]Department of Computer Science and Engineering and Information Technology,
School of Technology, Assam Don Bosco University, India
[3]School of Computing, Teesside University, UK
Email: [1]eshabarlaskar@gmail.com, [2]jayanta@dbuniversity.ac.in, [3]bissac@ieee.org

Abstract: **In order to enhance resource utilization and power efficiency in cloud data centers it is important to perform Virtual Machine (VM) placement in an optimal manner. VM placement uses the method of mapping virtual machines to physical machines (PM). Cloud computing researchers have recently introduced various metaheuristic algorithms for VM placement considering the optimized energy consumption. However, these algorithms do not meet the optimal energy consumption requirements. This paper proposes an Enhanced Cuckoo Search (ECS) algorithm to address the issues with VM placement focusing on the energy consumption. The performance of the proposed algorithm is evaluated using three different workloads in CloudSim tool. . The evaluation process includes comparison of the proposed algorithm against the existing Genetic Algorithm (GA,) Optimized Firefly Search Algorithm (OFS), and Ant Colony (AC) algorithm. The comparision results illustrate that the proposed ECS algorithm consumes less energy than the participant algorithms while maintaing a steady performance for SLA and VM migration. The ECS algorithm consumes around 25% less energy than GA, 27% less than OFS, and 26% less than AC.**

Keywords: - Virtual Machine Placement; Metaheuristic algorithms; Enhanced Cuckoo Search Algorithm; Cloud computing

I. Introduction

Cloud computing delivers a pool of on-demand computing resources over the Internet. The cloud data centers use virtualization technology (VT) to enable the resources of a single large server to be divided into several isolated execution environments running on numerous performance-isolated platforms known as virtual machines (VM) [1,2]. Recent research shows that enormous amounts of energy are consumed because of the incompetent usage of the resources. The fully ideal servers devour about 70% of their peak power [3] and are under-utilized [4]. The VT supports the datacenters to run with fewer physical servers, optimizing the usages of server and hence reduces the cost of the hardware and operation. The physical resources are shared dynamically in CC environment allowing running various applications on VMs on one physical machine (PM). Dynamic VM consolidation method gives better energy efficiency in Cloud data centers [5]. Thus, the information technology (IT) companies can save the costs of buying IT infrastructure and on maintenance.

The study of the dynamic VM consolidation will find solutions for the following issues.

1. To determine a PM is being overloaded or not. If yes, it will adopt the overloading detection algorithms to perform migration to migrate to other VMs;
2. To determine a PM is being under-loaded. If under-loaded, it will adopt under-loading detection algorithms to initiate the process of bringing the PM to a sleep mode by migrating all VMs from it;
3. To determine which VMs must be selected to migrate from the overloaded physical machine (VM selection) and

4. To determine which physical machines must be chosen to place migrated VMs (VM placement) [6].

The general challenge is to minimize the energy usage, operational cost and carbon dioxide emission. It aims to solve the issues in VM placement in Infrastructure as a Service (IaaS) by finding better methods to provide a solution for above challenges. The Cuckoo Search (CS) algorithm has three main components: selection of the best, exploitation by local random walk, and exploration by randomization via Lévy flights globally. Most metaheuristic algorithms use uniform distribution to generate new explorative moves. If the search space is large, Lévy flights are usually more efficient. A suitable combination of the above three components can thus lead to an efficient algorithm. The CS algorithm with Lévy flight locates the best solutions from the current solutions in the entire list of PMs by searching locally and globally at the same time for the optimal solution. It makes sure that the system will not be trapped in a local optimum. It is found that numbers of parameters to be tuned is less than those of Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) and adapts to a wider class of optimization issues [7]. The study proposes a hybrid cuckoo search algorithm named as Enhanced Cuckoo Search (ECS) which can consume less energy during VM placement while decreasing both SLA violation and VM migration**.**

It is important to have knowledge on the characteristics of VMs and PMs, for example on processing elements, energy usage by CPU, the status of memory, and bandwidth, etc. The amount of loading of VM instance varies with time. If the loading crosses the upper threshold value of a PM, it makes system imbalanced. To avoid such problems, VM allocation must be performed efficiently by enhancing the resource utilization. In Cuckoo Search algorithm, agents representing the cuckoo's travel around the search space and record the best solutions and have capacities to map the VMs into available PMs by optimizing the energy usage. So the problem is to design an efficient algorithm which can consume less energy during VM placement along with minimizing both SLA violation and VM migrations. The study will consider the concepts of the following: (1) VM placement with Lévy flight, (2) algorithms for overload detections, (3) policies of VM selection, (4) use of Status Index (SI) for CPU utilization prior to a VM selection. These strategies will help to resolve some of the issues that may rise due to heterogeneous nature of the cloud.

To detect the system overload, this study uses different measures such as Static Threshold (THR), Median Absolute Deviation (MAD), Inter Quartile Range (IQR), Local Regression (LR) and Robust Local Regression (LRR). Each host occasionally executes an overload detection algorithm to avoid performance degradation and SLA violation. Some concept of the algorithms is discussed below but the details are provided in [6].

1. A Static Threshold (THR) algorithms works on a situation where CPU utilization threshold value detects a host overload.
2. The Median Absolute Deviation (MAD) is a measure of statistical dispersion and it is considered as a robust estimator.
3. Inter Quartile Range (IQR) sets adaptive CPU utilization threshold based on another robust statistic, like the difference between the upper and lower quartiles
4. Local Regression (LR) works for fitting models to localized subsets of data to build up a curve that approximates the original data.
5. Robust Local Regression (LRR) works similar to LR but with extra robustness weight.

Once the overloads are detected, it uses the different policies of VMs selection such as Maximum Correlation (MC), Minimum Migration Time (MMT), Minimum Utilization (MU) and Random Selection (RS) [6]. A brief explanation of the VM selection policies are given below, however the details are provided in [6].

1. Minimum Migration Time (MMT) chooses the VM that requires the minimum time to complete a migration relatively. The migration time is estimated as the amount of RAM utilized by the VM separated by the spare network bandwidth available for the host.
2. Random Selection (RS) selects a VM to be migrated from the host according to a uniformly distributed discrete random variable.
3. Maximum Correlation (MC) selects VMs that have the highest correlation of the CPU utilization with the other VMs.

The study of Fan et al. [3] found a strong relationship between the CPU utilization and power consumption. This study introduced a status index (SI) to record the situation of VMs based on their CPU utilization pattern: (1) Underutilized VMs: usage below 30%, (2) Nearly overloaded VM: usage above 60% and (3) Normal load: usage 31% to 59%. A sample diagram is given in figure 1. The index values are updated on every new assignment, VM migration and completion of a task. The statuses of VMs of each PM are maintained by a local manager and the global manager maintains the status of all PMs. On arrival of a new VM instance, it tries to map to the most appropriate (underutilized) VM avoiding a random search. Thus SI helps to minimize the migration time of VMs.

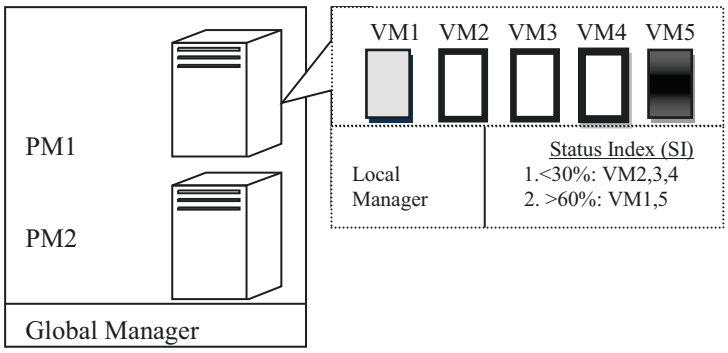

Figure 1. Local Manager with Status Index that keeps record of CPU utilization percentage of VMs

Thus the ECS algorithm adopts the combined merits of (1) algorithms for overload detections, (2) policies of VM selection, and (3) use of Status Index (SI) for CPU utilization prior to a VM selection. The results of ECS are compared with the results of the Genetic algorithm (GA), Optimal Firefly Search (OFS), and Ant Colony (AC). Analysis reports are provided to show the correlation between the impact of chosen algorithms for overload detection and policies of VM selection. The experimental results show that the proposed ECS algorithm consumes less energy with minimum numbers of SLA and VM migration.

The rest of the paper is organized as follows. The section 2 describes the related work. In section 3 the problem formulation of VM Placement is explained. Then the proposed methodology for VM placement along with the advantages of Cuckoo search algorithm is discussed in section 4. In section 5 the experimental results of the evaluation of proposed algorithm are provided. The section 6 concludes the study.

II. Related Works

In this section we present the review of the existing related research work. The related work is divided into three subparts: first we present a review of a few existing VM placement algorithm followed by a review of different areas where cuckoo search algorithm has been used and finally the concepts of different meta-heuristic algorithms used in this paper are explained.

A. A Few existing VM placement algorithms

VM placement is a crucial few applications of Cuckoo Search algorithm approach to better resource utilization and energy efficiency in CC infrastructures. Various research works have pontificated the significance of the VM placement problem relevantly. For instance M. Cardosa et. al. [8] have described the importance of placing VM into PM appropriately. The VM placement problem is an NP-hard problem which is generally constructed or done as an alternative to the vector bin-packing problem as per [9]. In [6] the authors proposed a Power Aware Best Fit Decreasing (PABFD) algorithm for VM placement which is a modification of Best Fit Decreasing algorithm (BFD). The authors in [10][11][12][13][14] have also formulated numerous heuristics for VM placement problem. For instance, a novel approach for VM clustering was introduced that uses the mixture of Gaussians (MoGs) with the Kullback-Leiber divergence to model similarity between VMs[10]. The study in [11] the authors dealt with the tradeoff between cost and power dependent on tight performance constraint by packing as much VMs in a small number of physical machines and this reduced the cost of VM migration. The author in [14] designed a single-objective algorithm based on max-min ant system (MMAS) metaheuristic to reduce the total amount of PMs needed to handle the currently available load. Some metaheuristic algorithm is also widely used for solving the problem of VM placement, such as the genetic algorithm (GA), honeybee algorithm (HB), ant colony optimization algorithm (ACO) etc. In [16] the author proposed GA as a scheduling strategy for load balancing of VM resources. The cloud computing uses two types of scheduling. The first type is task scheduling where tasks are mapped to a virtual machine (VM). The second type is VM scheduling which is also known as resources scheduling. The resource scheduling in Cloud considers only the current state of the system but hardly considers system variation and historical data and as a result of which load imbalance of the system occurs. Therefore, to tackle this problem in VM resource scheduling the authors used GA by considering historical and current data as inputs to the system. This method helped in computing in advance the influence on the system once the required VM resources were deployed and then picks the least effective solution which resulted in best load balancing and at the same time avoided dynamic VM migration thereby reducing the cost of migration.

In [16] Shailesh Sawant proposed a GA based VM resource scheduling strategy that focuses on system load balancing. The study is similar to the work done in [15] in which the GA approach finds the effect of the deployment of new VM resources in the system . The author proved that the traditional algorithm, when used for resource scheduling, ends up in an imbalance of load and the number of VM migration also increases. In [17] another GA based approach (GABA) was proposed which could self-reconfigure the VMs in CC data centers consisting of heterogeneous PMs. In [18] the VM placement problem is designed as a multi-objective optimization problem to minimize various issues such as power consumption, resource wastage and the cost of thermal dissipation. To tackle all these issues, the authors proposed an optimal GA with fuzzy multi-objective evaluation. In [19] the authors proposed eco-friendly algorithm by combining both honey bee and ant colony algorithm for cloud computing which reduced the operational cost by minimizing power consumption which in turn also diminished global warming to a great extent. The proposed Bee-Ants colony system was used for proper energy efficient resource management where initially the jobs are divided into two parts. The first part looks after the proper management of overloaded. The underloaded CPUs with service rescheduling was carried out by honey bee algorithm. The second part which helps to manage the idle CPUs (power consumption management) is achieved by ant colony algorithm. In [20] a multi-objective ant colony system algorithm was proposed for the VM placement with the aim of obtaining a group of non-dominated solutions that manages the tradeoff between resource wastage and power consumption. The authors compared the proposed algorithm with multi-objective GA, two single-objective algorithms namely bin packing and MMAS. The outcome of the experiment proved that the proposed algorithm is much efficient than the algorithm it was compared to. The advantages of packing VMs competently in the consolidation of servers is

described in [21] by W. Vogels. The authors in [22] and [23] explain the placement problem based on proxy method while the authors in [24] and [25] describe the object placement/replacement for simple replication of data. All these works [22],[23],[24],[25] aim to impose upon the elasticity that can be obtained in deciding suitable placement. A novel virtual machine scheduling approach was introduced in which the scheduler allows virtual machines to obtain extra CPU shares [26].

A proposed efficient algorithm which was established in linear and quadratic programming helps in making the placement of VMs on PMs at optimum level by minimizing the usage of the total number of nodes [27]. The server consolidation problems with the formulations of linear programming extended restrictions for the problem of VM allocation[28]. The restriction was that the VMs allocated to a PM should be based on some unique attribute so that the total number of VM migrations can be minimized and also a heuristic based on LP-relaxation was built to optimize the linear program solving cost. The energy aware resource allocation in grid computing is also discussed in [29]. In [30] and [31] the authors used the method of constraint programming for VM placement problem. The authors in [30] addressed the problem of VM provisioning and placement as two constraint satisfaction problem and they proposed a framework for resource management by combining dynamic VM provisioning manager and VM placement manager which are utility based. On the other hand, the authors in [31] solved the constraint programming based dynamic consolidation problem by designing an entropy resource manager for similar clusters which considers both the issues of VM allocation and VM migration to the available nodes. The scheme of controlling the load of task in order to decrease the total cost of task migration was studied in [53].

B. A Few applications of Cuckoo Search algorithm

One of the newest nature-galvanized metaheuristic algorithms is the Cuckoo search (CS) algorithm, developed in 2009 by Xin-She Yang and Suash Deb [33],[34],[35]. Brood Parasitism (BP) method is adopted by some species of cuckoo. BP is a procedure of reproduction observed in birds that implicate the laying of eggs in some other birds' nests wherein the eggs are left under the parental protection of the host parents (often between other species which is called inter-specific or within a species which is called intra-specific). Few species of cuckoos, for instance, the Ani and Guira cuckoos lay their eggs in neighborhood nests and also they sometimes may remove others' eggs to raise the hatching possibility of their own eggs. In CS algorithm, the method of BP is used and also this algorithm is intensified by the so-called Lévy flights [36] instead of the simple isotropic random walks. CS is found probably to be far more efficient and competent than GA and PSO (Particle Swam Optimization) [33]. Cuckoos are known not only for their delightful sound but also for their combative approach to reproduction.

Three idealized rules have been used in order to describe the standard CS with ease [37]. The rules are as follows: A nest is chosen at random. Each cuckoo lays one egg at an instant. The best solutions are represented by best nests with good quality eggs in it. These set of solutions are then carried over to the next generation. Hosts nests available are fixed. Whenever a cuckoo's egg is discovered by a host then they are considered as worst solutions. They are dumped and not taken further to the next generation. A particular host can discover a cuckoo egg with a probability of $pa \geq (0, 1)$. In this case, the host bird can either slaughter or throw away the egg or simply relinquish and vacate the nest and build an entirely new nest.

CS is a pleasant and efficient amalgamation of PSO, differential evolution (DE) and simulated annealing (SA) in one algorithm. Therefore, DE, PSO, and SA can be deliberated as a special condition of CS. Two types of searches can be performed by CS, namely, local search and global search. These two searches are regulated and supervised by a switching/discovery probability. Usually the local search is very exhaustive with about one-fourth of the search time whereas three fourth of the total search time is taken by the global search. As a result of this the entire space can

be explored by the global search process in the much efficient manner and, therefore, the probability of finding the global optimality is very high. One more benefit of CS is that it uses Lévy flights for global searches instead of just taking standard random walks and because Lévy flights have infinite mean and variance, it helps significantly in exploring the search space. This advantage along with local search capabilities and global convergence makes CS very effective and efficient. The efficiency of CS has been shown in various research studies and applications in different fields [34][38][39][40][41][42].

C. Explanation of different meta-heuristics algorithms which are compapred in this work

1. Concept of Cuckoo Search (CS) Algorithm

The CS concept has been taken and inspired from the brood parasitism of cuckoo birds. Cuckoo birds never build their nests. They tend to lay their eggs in the nests of other birds. The breeding behavior of this nest searching technique can be used for various optimization problems. Each solution is represented by an egg in the nest,. The new egg represents a new solution. The aim of this searching technique is to replace a not-so-good solution in the nests by a comparatively better solution. Its simplest form can be taken as one egg in a nest. The algorithm can be extended to more complicated cases with multiple eggs in multiple nests.

In CS algorithm, the action of cuckoos is connected with Lévy flights so as to obtain a new nest in an efficient manner. A French mathematician Paul Lévy discovered Lévy flights that denote a pattern of random walks distinguished by their step lengths which follow a power-law distribution. This pattern is usually presented by small random steps adopted in the long term by huge jumps [33][43][44]. CS is one of the most newly designed metaheuristic algorithms developed by Yang and Deb and the details are given in their study report [47]. Some principles and rules are given below.

A nest is chosen at random. Each cuckoo lays one egg at a time. The best solutions are represented by best nests with good quality eggs in it. These set of solutions are then carried over to the next generation. Hosts nests available are fixed. Whenever an egg is discovered by a host, they are considered as worst solutions. They are dumped and not taken further to the next generation. A host can discover a cuckoo egg with a probability of pa $\geqslant$ (0, 1). In this case, the host bird can either slaughter or throw away the egg or simply relinquish and vacate the nest and build an entirely new nest. The Lévy flights (Lévy (λ)) is given as in equation number 1. The global random walk of Le'vy flights is given as in equation number 2.

The way in which exploitation and exploration of the solution space as performed by a cuckoo is considered to be the strength of CS. Some kind of intelligence is used by this cuckoo to search for a more effective and efficient solution. In [45] it was seen that the cuckoos engross themselves in some kind of vigilance where they find the much better nests and as a result of this a new class of cuckoos can be created that have the capability to change the host nest at the time of evolution and to refrain from forsaking of eggs. The cuckoos observe the host nests very carefully before the brooding process, and once it finds a nest which is the best choice at that point of time, then it chooses that particular nest. Thus, it can be said that a fraction of cuckoos performs a local search around current solutions. This new fraction of cuckoos having being influenced by the observed behavior take up two crucial procedures, which are as follows: Firstly, by using Lévy flights a cuckoo move towards a new solution which denotes a new area. Secondly, the cuckoo can search for a better solution from the current solution wherein it performs a local search.

There are three types of cuckoos that can be fabricated from the entire population of cuckoos, which are given below: A cuckoo searching in different areas for new solutions obtains a better solution

from their current best position. These set of cuckoos can be chosen randomly from the entire population. A portion (pd) of cuckoo looks for new solutions distant from the best solution. Another portion (pp) of cuckoo explores new solutions from the current best position and attempt to enhance them. These cuckoos proceed from one place to another by using Lévy flights to find the best solution in each place without being caught in the trap of local best.

The solutions found in the local search and the solutions found at distant from its current best solutions are the key factors that guide the entire population of cuckoos in their search process. This process enhances exhaustive search around numerous best solutions, and simultaneously randomization is also performed effectively using Lévy flights for finding new areas. In the expansion of standard CS, the search process was performed in fewer iterations and giving better defiance to any promising traps and inactivity in local optima. Thus it becomes more efficient. The new method described that from each of the solution, by exploring other options using Lévy flights, the cuckoos attempt to locate the best solution in a region by searching at the local level.

2. Concept of Optimal Firefly Search Algorithm (OFS)

Xin-She Yang has developed firefly search algorithm in late 2007 and 2008 at Cambridge University [48]-[49]. In this algorithm, the flushing characteristics of fireflies were projected in the form of algorithm where three main assumptions were made. Inspired from the FA algorithm a new OFS algorithm is designed where the assumptions were modified.

In OFS algorithm, the first assumption was made that the VMs are the female fireflies and PMs are the male fireflies unlike the assumption made in original FA where the fireflies are considered to be unisex. The male firefly will be attracted towards female firefly depending on the brightness. If the PM has less brightness, then it will be attracted towards brighter VM. The brightness is considered to be more if the PM/VM is not overloaded or slightly loaded, and the brightness is less if it is overloaded.

In the original FA it is considered that attractiveness and brightness are proportional to each other. In the second assumption the same is also assumed in the OFS and if there is a less bright VM and if two PMs are available then that VM will be attracted towards the PM with more brightness. Also as the distances increase the attractiveness and brightness decreases. In this case when the resource utilization of both the PM and VM increases then only the distance between them will increase.

The third assumption is depicted in the view of an objective function. In OFS, the brightness of male and female fireflies are determined by the objective function which is the resource utilization of the PMs and VMs. A VM/PM will have more brightness if the use of resources is in between the lower and upper threshold values. If a PM is having the use of resources below the lower threshold then it will be considered as less bright and even if the resource utilization is above the upper threshold, then also the PM becomes less bright. From the above assumptions, the OFS algorithm was applied to solving the problem of VM placement in Cloud Data Center.

3. Concept of Genetic Algorithm for VM Placement [16]

A genetic algorithm is a heuristic based search technique. It is particularly useful in problems where objective functions dynamically change. Typically, a genetic algorithm starts with a population of the solution, applies genetic operators on this which results in an optimal solution [50]. A variation of a genetic algorithm known as grouping genetic algorithm can also be applied to the VM placement problem. These algorithms can take into account additional constraints while optimizing the cost function. This is particularly useful in cases where we need to operate on groups [52].

The genetic algorithm mainly involves the following aspects:

(a) Chromosome modeling - This is an encoding schema used to encode details of the problem that is to be passed from one generation to the other.
(b) Population Initialization - The feasibility of the solution is dependent on the feasibility of the initial solution.
(c) Crossover - It is a genetic operator used to vary the programming of a chromosome from one generation to next.
(d) Mutation - Mutation is a genetic operator that alters one or more gene values in a chromosome from its initial state. This helps to prevent the population from stagnating at any local optima.
(e) Generation Alternation - This is where we select one of the solutions from the next generation of solution

4. Concept of Ant colony algorithm

The concept of the multi-objective ant colony algorithm [51] was simulated in the CloudSim environment. The solution generated by the ant colony algorithm is a permutation of VM assignment. The algorithm works in a number of stages. The first is the initialization stage where parameter values are set, and the values of pheromone trails are also set. Next the VM requests are sent to each ant which is then assigned to the physical hosts by using a rule that is pseudo-random. This rule is dependent on a heuristic that supervises the ants in choosing the most favorable VMs and also the information about the concentration of the current pheromones. All the artificial ants will build their local pheromone updates and then a global update is performed with each solution of the current solutions.

Also by considering the outcome of multi-objective ant colony algorithm it is observed that the algorithm in [51] is compared with other existing algorithms with respect to performance and scalability by setting up simulation experiments with programs in Java as in [51]. However, the algorithm was simulated on homogeneous server environments where a solution was randomly chosen from the several non-dominated set of solutions. It has been observed that the multi-objective ant colony algorithm performed significantly better than some other algorithms to which it was compared.

III. Problem Formulation for VM Placement

A. VM placement with Lévy flight

Consider a set of VMs represented by VM= {vm_1, vm_2, …., vm_n} to be placed on a set of heterogeneous physical machines_(PMs). The VM are represented as vm_i = (PEj, CPUi, RAM_i, BW_i), $1 \leq i \leq n$. The values PEj, CPUi, RAM_i, BW_i denote processing elements (PEj), Power consumption by CPU (CPUi), MBytes of physical memory (RAM_i), and Kbits/s of network bandwidth (BW_i) respectively. Some key parameters of the study and system model are motivated from an earlier study [6]. Let PM = {pm_1, pm2,.., pm_m} denote a set of PMs. Each PM are represented as pm_j = (pe_j, cpu_j, ram_j, bw_j), $1 \leq j \leq m$. The study concerns more to three types of computing resources such as processors, physical memory, and network bandwidth. The value of pe_j, cpu_j, ram_j, bw_j denotes the total resource capacity of the j^{th} PM. In addition, x_{ij} , $1 \leq i \leq m$, $1 \leq j \leq n$ and y_i, $1 \leq i \leq m$ are decision variables, x_{ij} = 1 if and only if vm_j is mapped onto pm_i, y_i = 1 if pm_i is used to host virtual machine. The objective function is to minimize $\sum_{i=1}^{m} y_i$ while finding all values of x_{ij}.

There are few constraints. For each type of resources (CPU, memory and bandwidth), the quantity of resource requests of VMs placed in the same physical machine must be less or equal to

ability/capacity of the PMs hosting them. The total numbers of PMs that allocate VMs are not more than a specified limit [6]. The CS algorithm with Lévy flight locates the best solutions from the current solutions in the entire list of PMs by searching locally and globally at the same time for the optimal solution [7]. It thus makes sure that the system will not trap in a local optimum. It is found that numbers of parameters to be tuned is less than those of Genetic Algorithm and Particle Swarm Optimization (PSO) and adapts to a wider class of optimization issues. The CS has the ability to solve non-convex, nonlinear, non-differentiable, and multimodal problems [33].

For example let us say a cuckoo 'i' produce a new solution $x_i^{(t+1)}$ to a given problem from a known solution x^t at time t. Then a Lévy flights (Lévy (λ)) is performed (equation 1)

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus \text{Lévy}(s,\lambda) \quad \ldots \quad (1)$$

Where, $\alpha > 0$ is a step size scaling factor, which should be related to the scales of the problem of interest, and s is the step size. The product ⊕ means entry-wise multiplication. The global random walk carried out by using Le'vy flights has an infinite variance with an infinite mean (equation 2)

$$\text{Lévy}(s,\lambda) \sim s^{-\lambda}, (1 < \lambda \leq 3) \quad \ldots \quad (2)$$

The metrics and model are as follows:

(i) CPU utilization

Let F be the clock frequency to model the capacity of a single core CPU. The CPU utilization of a VM, *ui* is relative to the VM's CPU frequency *fi* which is a fraction of the host CPU utilization *U*. Then the host CPU utilization can be taken as summed up over the N VMs allocated to the host (as given in equation 3). For a multi-core CPU with *n* cores *F* will be replaced by *nf*.

$$U = F\sum_{i}^{N} fiui \cdots \quad (3)$$

(ii) The Cost of VM Migration

During migrations the VMs allows transferring a VM between physical nodes without suspension. The study of Voorsluys et al [47] estimated the amount of performance degradation is approximately 10% of the CPU utilization. It may cause an SLA violation. The duration of a live migration depends on the total amount of memory used by the VM and available network bandwidth. The performance degradation and migration time experienced by a VMj are estimated as given in equation (4) and (5) respectively

$$U_{d_j} = 0.1.\int_{t_0}^{t_0+T_{m_j}} u_j(t)dt \quad \ldots \quad (4)$$

$$T_{mj} = \frac{M_j}{B_j} \quad \ldots \quad (5)$$

Here t_0 is the start time of migration, T_{mj} the time taken to complete the migration, $u_j(t)$ the CPU utilization of VMj, m_j is the memory used, and Bj is available bandwidth.

(iii) SLA violation metrics

The SLA violation can be defined by two metrics [6] as follows: 1) SLA violation time per active PM that rises with overload time period of the PM as given in equation 6. Let's call this as SLA with overtime (SLAOT). 2) Performance degradation due to migrations (PDM) as given by equation 7.

$$SLAOT = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}} \quad \ldots \quad (6)$$

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{D_{d_j}}{D_{r_j}} \quad \cdots \quad (7)$$

where N and M are the numbers of host and VMs respectively; T_{s_i} is total time during which physical machine *i* has experience maximum CPU utilization; T_{a_i} is total time during which physical machine *i* is serving VMs; D_{d_j} is an estimation of the performance degradation of the VM_j caused by migration; D_{r_j} is total CPU capacity requested by VM_j during its lifetime. A metric for describing SLA violation can be defined by equation 8:

$$SLAV = SLAOT \times PDM \quad ... (8)$$

(iv) Overall energy consumption

The estimated combined metric is that captures both energy consumption (E) and the level of SLA violations. Let's call this term as Energy with SLA (ESLA) as given in equation 9.

$$ESLA = E \times SLAV \quad \ldots \quad (9)$$

B. Use of Status Index (SI) for CPU utilization prior to VM selection is done as per the following rules

The study of Fan et al. [3] found a strong relationship between the CPU utilization and power consumption. The status index (SI) periodically updates the status of VMs based on their CPU utilization pattern as follows: (1) Underutilized VMs (2) Nearly overloaded VM (3) Normal load. A sample system is shown in the figure 1. The index is updated on every new assignment, VM migration and completion of a task, etc. The status of VMs of each PM is maintained by a local manager and the global manager maintains the status of all PMs.

Sample generic rules are as follows:

1. Assigned status Index (SI) value of VM to 1, if the CPU utilization of those VM is less than 30%
2. Assigned status Index (SI) value of VM to 2, if the CPU utilization of those VM is more than 60%
3. Assigned status Index (SI) value of VM to 3, if the CPU utilization of those VM is between 31 to 59%
4. Sort the values of the VMs in ascending order (less utilized VMs will be on top of the index.

On arrival of a new VM instance, it tries to map to the most appropriate (or underutilized) VM avoiding a random search. The SI values helps to handle the different loads dynamically. Thus SI helps to minimize the migration time of VMs by avoiding migrations to a great extent.

IV. Proposed Methodology

In the recent years, the advancement of nature-inspired metaheuristic instigated researchers to implement metaheuristic for resolving various combinatorial problems. For VM placement issues in cloud data centers, the conspicuous growth in the size of the solution search space inspired the researchers to apply nature-inspired metaheuristic to handle the VM allocation problems. In this section, we introduce our proposed Enhanced Cuckoo Search algorithm.

A. Concept of the Cuckoo Searches in VM Placement Problem

In refitting the CS concept to VM placement problem, the proper translation of terminology used in the CS must be done efficiently and is the crucial factor in a combinatorial space from a regular one. VM placement is one of the combinatorial optimization problems and as such the key concepts related to CS must be described in accordance with VM placement problem before solving this problem. The basic CS algorithm assumes that in one nest only one cuckoo can lay one egg at a time. By modifying the concept, let us assume that in one nest at least two eggs can be laid by one or more cuckoos. This assumption is made due to the fact that the PMs are dual cores, and one core can be assigned to one VM. Therefore, at most two VMs can coexist in a PM at a time. To familiarize CS to VM placement, the five crucial elements need to be discussed, namely, egg, nest, objective function, search space and Lévy flights. Considering cuckoos to be single agents for mapping VMs into PMs, the following assumptions are made:

1) Egg:
   Consider that a cuckoo can lay two eggs in one nest. An egg in a nest is a solution depicted by one VM from the entire list of VMs. An egg can either be an existing VM or a new candidate VM requested by a new user or old users. Thus eggs in CS are equivalent to VMs in VM placement problem.
2) Nest:
   The number of nests is fixed in CS, which is the total population size. In the case of VM placement problem, a nest can be projected as a PM with its available resources. It is also assumed that a nest can hold only two eggs at a time since the PMs are considered to have dual-core processors. Obviously, a nest can have more than two eggs in future discussions. However, in this case only two eggs are considered to be placed in a single nest.
3) Objective Function:
   An objective numeric value is related with each of the solutions in the search space. Hence this objective function's value is directly proportional to the quality of the solution. In CS, an egg of better quality will be a part of new generations which indicates that the probability of having a new cuckoo from an egg is directly associated with the quality of the cuckoo's egg. In the case of VM placement problem, the quality of a solution is related to the optimal VM placement, that is, to allocate the VMs into PMs in an effective and efficient manner.
4) Search Space:
   In the case of CS, the search space represents the positions of promising nests and in order to change the positions of these nests the actual values of the coordinates need to be optimized. It is observed that in most of the continuous optimization problems the moving nests or locations of the nests have got no real constraints, which is particularly true for techniques where a solution is moved from one neighborhood to another. In VM placement problem, the search space corresponds to the list of PMs. The number of PMs is fixed and considered according to their resource availability to serve the VM requests.
5) Lévy flights:
   Lévy flights have a typical feature of an exhaustive search for a solution, followed by sporadic large steps in the long run. As reported by Yang and Deb [33], in some optimization problems the Lévy flights proved to be most efficient in the search for a new best solution. To enhance the quality of search, the step length will be connected to the value given by Lévy flights as stated in the basic CS. The search space (solution space) must have an idea/knowledge of steps, and precise and constant as considered in [46].

B. Proposed Enhanced Cuckoo Search (ECS) algorithm

In Cuckoo Search algorithm, cuckoos are assumed to be simple agents that travel and communicate around the search space and record the best solutions that they discover during their search process. Therefore, CS can be used to generate solutions for placement of VMs so that the VM

placement/allocation process is implemented efficiently, and the mapping of VMs into available PMs is performed in this way that it reduces energy consumption while not violating SLAs. The important steps of CS algorithm are locating the best solutions from the current solutions with the CS exploration process using Lévy flights by a random step from the current solution (that is the current PM) and to attempt to locate the best solution (most suitable PM) in the entire list of PMs by searching locally. The last step is evaluating the fitness function and generating the new list of placements for VMs. The most critical issue in generating a successful solution is the proper depiction of the optimization methods. Such algorithm can perform efficiently even in a large scale, dynamic cloud setting because of its capability to utilize Lévy flights to produce new solutions (eggs) as discussed in the equation 1 and 2 [54]. In the proposed enhanced CS algorithm, each cuckoo is considered as a vector of natural numbers where the elements represent the PMs to which the VMs are to be mapped. The new enhanced CS algorithm is given below:

Algorithm: Enhanced Cuckoo Search for VM placement

The PM is the physical machine; VM is the virtual machine; resources are the processing element and are given by the combination of three different types of resources like the number processors in each PM, the speed or efficeincy (million instructions per second) of all processors in each PM and the bandwidth or communication ability of each PM.

Input: pmList, vmList
Output: allocation of VMs
1: Objective function f (x), x = (x1, . . . , xd)T, where the objective function f(x) gives set of physical machines which are available for VM placement
2: Generate initial population of n host nests xi (i = 1. . . n) which represents the initial list of physical machines
3: Generate the values of the Status Index (utilization % of nests)
4: While all the VM requests are allocated/ served, or no PM is available do
5: Start searching with the type of resource requests for VMs of smart cuckoos/users
6: Get a cuckoo randomly by Lévy flights
7: Choose a nest among n (say, j) based on minimum utilization of Status Index
8: If overload detected, use an effective nests election policy
9: Evaluate its quality/fitness Fi; i.e. the amount/ type of resources requested by the VM and whether such requests can be served or not.
10: Choose a nest among n, say j.
11: if (Fi > Fj), i.e. if the resource requirements by VMi is more than the available resources provided by the PMs of the jth VM), then
12: replace j with the new solution;
13: end if
14: A portion of worse nests (pa) is replaced by new nest
15: Calculate the quality of nests and keep the nests
16: Rank the solutions and find the current best
17: End while
18: Post processing of results

C. Some advantages of Cuckoo search algorithm

The proposed Enhanced Cuckoo Search (ECS) algorithm was compared with other existing bio-inspired algorithms along with the newly proposed Firefly algorithm. From the results of the comparison, the following advantages of Cuckoo search algorithm have been found.

The cuckoo search satisfied and guaranteed the global convergence requirements. The ECS is used for resource management during VM placement in all the cloud settings irrespective of their nature (private, public or hybrid). The cuckoo search has two search capabilities: local search and global

search, controlled by a discovery probability. The local search is very intensive with about 1/4 of the search time while a global search takes about 3/4 of the total search time. It allows the search space be explored more efficiently on the global scale, with higher probability. The cuckoo's global search uses Lévy process instead of standard random walks. Due to the Lévy process, ECS can explore the search space more efficiently. This advantage combined with both local and search capabilities and guaranteed global convergence, makes the cuckoo search very efficient. The nature-inspired metaheuristic algorithms perform powerfully and effectively in solving the diverse optimization problems with the combinatorial optimization problems.

In CS algorithm, cuckoos are assumed to be agents that travel around the search space and record the best solutions. The proposed algorithm generates solutions for placement of VMs efficiently. It maps VMs into available PMs optimizing the energy uses without compromising the SLAs. After this for possible replacement of a solution, all of the nests are ranked by fitness and the worst fraction of the nests is replaced with random solutions. It also locates the best solutions from the current solutions using Lévy flight in the entire list of PMs by searching locally. Then it evaluates the fitness function and generating the new list of placements for VMs. In the proposed enhanced CS algorithm, each cuckoo is considered as a vector of natural numbers where the elements represent the PMs to which the VMs are to be mapped. This combination of mechanisms allows the solutions to search locally and globally at the same time for the optimal solution.

The deterministic search approaches have the drawbacks of being trapped into local minima unavoidably. The metaheuristic methods can deliver satisfactory solutions in a reasonable time. The CS does not use gradient information during the search so that it can solve non-convex, nonlinear, non-differentiable, and multimodal problems.

V. Implementation and Experimental Result

The proposed ECS algorithm for VM placement puts as many VMs as can be accommodated in a single PM after considering the required resources. This study focuses on energy consumption issue. We have used the Planetlab Workload traces which are available with the CloudSim toolkit [31] package to perform our simulation which is run for 24 hours with proposed VM placement algorithm along with the other existing overload detection and VM selection algorithms [6]. The power consumption data is collected from real data on power consumption provided by the results of the SPEC power benchmark [32]. The proposed Cuckoo search algorithm for VM placement have been implemented with four different VM selection policies like Maximum Correlation (MC), Minimum Migration Time (MMT), Minimum Utilization (MU), Random Selection (RS) and five host overload detection algorithms like Static Threshold (THR), Median Absolute Deviation (MAD), Inter Quartile Range (IQR), Local Regression (LR) and Robust Local Regression (LRR) [6].

We have simulated a data center that comprises of 800 heterogeneous physical nodes, half of which are HP ProLiant ML110 G4 servers, and the other half consists of HP ProLiant ML110 G5 servers. Each server has 1 GB/s network bandwidth. The characteristics of the VM types correspond to Amazon EC2 instance types. The details of the key parameter values and system model are motivated from an earlier study [6].

The characteristics of three different workloads are presented in Table 1, and the experimental result is shown in Table 2. The result of the comparison between workload 1, workload 2, and workload 3 is illustrated as in following figures. As observed the cuckoo search algorithm performs better with fewer VMs. If we keep on increasing the number of VMs as in workload 2 and 3, then the consumption of energy increases. It can be seen that the minimum energy consumed was in workload 1 and then in workload 3, and highest was in workload 2.

Table 1. Characteristics of some random workload Data

Data	Number of VMs	Mean	St. dev	Quartile 1	Median	Quartile 3
Workload 1	1052	12.31%	17.09%	2%	6%	15%
Workload 2	1516	9.26%	12.78%	2%	5%	12%
Workload3	1078	10.56%	14.14%	2%	6%	14%

Table 2: Energy consumption for different workloads by ECS algorithm, policies: overload detection and VM selection

ECS Algorithm using	Energy Consumption (kWh)		
Overload detection &VM Selection algorithms	**Workload 1**	**Workload 2**	**Workload 3**
IQR-MC Inter Quartile Range (IQR) Maximum Correlation (MC)	44.30	46.71	45.17
IQR-MMT Minimum Migration Time (MMT),	38.56	46.52	39.21
IQR-MU Minimum Utilization (MU)	34.32	41.31	33.30
IQR-RS Random Selection (RS)	37.29	43.29	33.91
LR-MC Local Regression (LR)	38.88	42.02	26.17
LR-MMT	32.08	33.99	35.09
LR-MU	*22.50*	34.50	33.05
LR-RS	34.75	36.75	32.79
LRR-MC Local Robust Regression (LRR)	39.54	40.52	33.66
LRR-MMT	32.14	35.17	28.16
LRR-MU	38.09	43.47	40.01
LRR-RS	36.62	39.71	33.32
MAD-MC Median Absolute Deviation (MAD)	33.28	37.27	32.73
MAD-MMT	32.54	38.84	32.91
MAD-MU	35.01	40.05	37.50
MAD-RS	41.22	46.84	43.98
THR-MC Static Threshold (THR)	33.09	38.88	36.76
THR-MMT	36.34	39.76	37.66
THR-MU	30.89	36.55	32.25
THR-RS	37.87	40.91	39.76

Figure 2, 3 and 4 shows the result of energy consumption of ECS algorithm with workload 1, 2, 3 respectively. It is shown that better results (less energy) are generated by LR-MU, LR-MMT, LR-MC etc for different loads (1,2,3). These are some potential policies that generate better result.

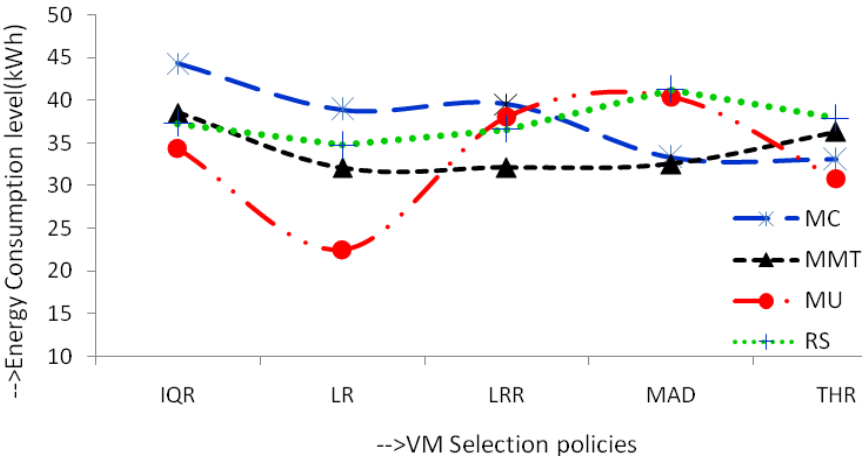

Figure 2. Energy Consumption of ECS algorithm with workload 1

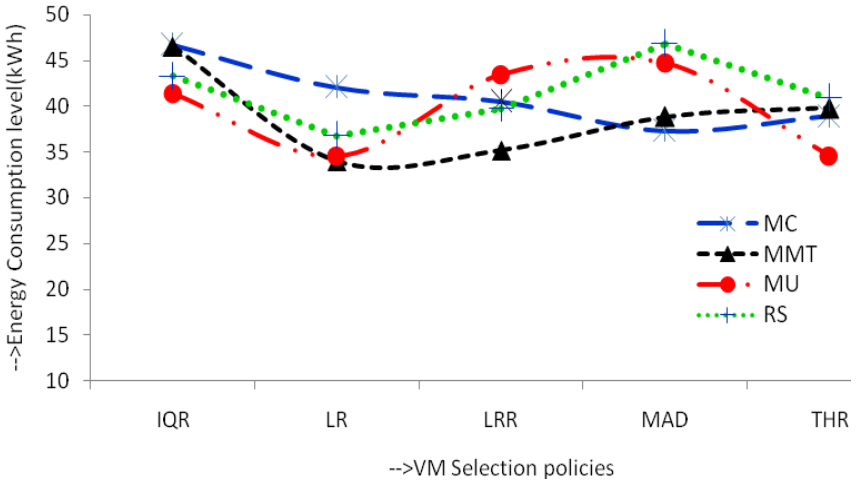

Figure 3. Energy Consumption of ECS algorithm with workload 2

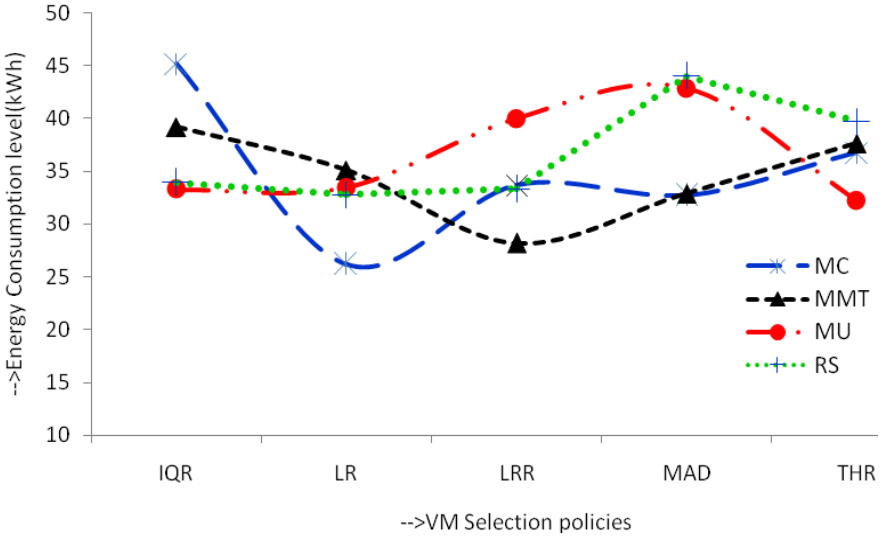

Figure 4. Energy Consumption of ECS algorithm with workload 3

Figure 5 gives the comparisons of energy consumption under different workloads by ECS algorithm. The average amount of energy consumed under different workloads by ECS algorithm are compared. On the average, VM selection policies like Minimum Migration Time (MMT) and Minimum Utilization (MU) gives better energy performance.

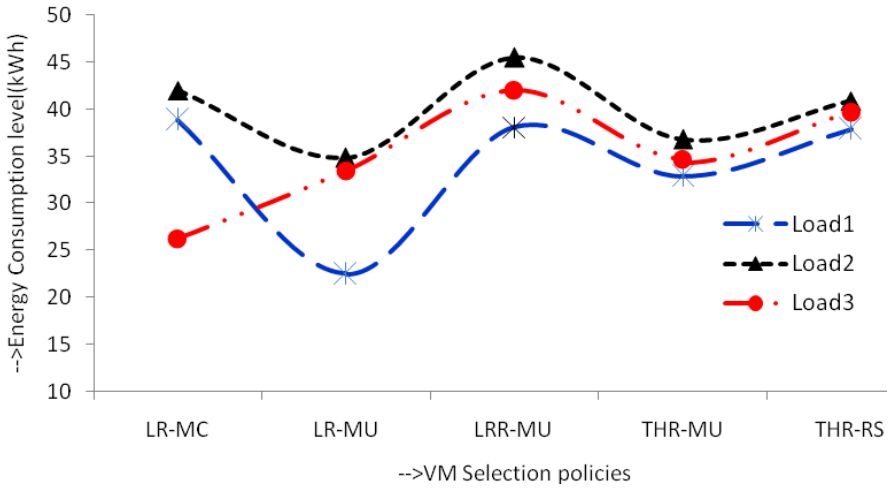

Figure 5. Comparisons of Energy Consumption of different workloads with ECS algorithm

We also explore more on contributing policies such as overload detection, VM selection and impact of Status index (for CPU utilization) with other algorithms. The detailed comparison of results of algorithms like Enhanced Cuckoo Search (ECS), the Genetic algorithm (GA), Optimal firefly search (OFS) and Ant colony (AC) are provided in table 3. From the values given in table 3 for the metrics, namely: energy consumption, SLA and number of VM migrations it is very clear that the energy consumption is significantly reduced in ECS algorithm as compared to the other three algorithms. The value of energy metric came down to 22.50 kWh from 44.30 kWh for ECS with LR-MU. Also, the lowest values for all three metrics among all the algorithms is shown by ECS which are 22.50 kWh energy consumption, 0.00007% of SLA violation and total number of VM migrated dropped to 822. Although, there is no substantial improvement in SLA violation and VM migration metrics but the proposed ECS algorithm performs efficiently in reducing metrics the total energy consumed. The reason for not having better results in SLA and VM migration is due to the fact that the total number of VM migrations are more while turning off the under-loaded PMs in order to reduce the overall energy consumed by the data center and as a result of which the SLA violation might also increase. However, as compared to GA, OFS and AC, the results of ECS seem to be promising.

Table 3: A Comparison of algorithms: Enhanced cuckoo search (ECS), Genetic algorithm (GA), Optimal firefly search (OFS), Ant colony (AC) as per the chosen policies of Overload detection and VM selection

Overload detection & VM Selection	**ENERGY**				**SLA**				**VM Migration**			
VM Placement	**ECS**	**GA**	**OFS**	**AC**	**ECS**	**GA**	**EFS**	**AC**	**ECS**	**GA**	**OFS**	**AC**
IQR-MC	44.30	41.05	32.17	32.88	0.00009	0.00009	0.00008	0.00009	876	820	869	844
IQR-MMT	38.56	32.00	32.21	31.80	0.00011	0.00010	0.00007	0.00008	854	810	880	809
IQR-MU	34.32	34.30	32.35	31.33	0.00010	0.00008	0.00009	0.00008	887	875	919	889
IQR-RS	37.29	34.23	32.91	30.63	0.00012	0.00008	0.00008	0.00007	882	810	867	877
LR-MC	38.88	35.45	31.81	32.32	**0.00007**	0.00009	0.00008	0.00006	886	890	907	878

LR-MMT	32.08	32.05	32.09	36.11	0.00008	0.00008	0.00007	0.00008	850	870	874	809
LR-MU	**22.50**	30.11	32.5	34.33	0.00008	0.00007	0.00008	0.00007	863	870	908	887
LR-RS	34.75	32.54	31.79	30.98	0.00010	0.00009	0.00007	0.00008	830	841	833	867
LRR-MC	39.54	40.08	31.66	33.92	0.00011	0.00009	0.00009	0.00011	876	878	923	829
LRR-MMT	32.14	33.60	30.87	35.86	0.00010	0.00008	0.00009	0.00008	863	867	971	912
LRR-MU	38.09	32.22	32.06	37.44	0.00011	0.00008	0.00008	0.00007	911	900	860	880
LRR-RS	36.62	35.50	32.32	34.58	0.00010	0.00008	0.00009	0.00008	**822**	830	871	899
MAD-MC	33.28	34.90	31.73	38.34	0.00010	0.00010	0.00008	0.00009	888	878	855	801
MAD-MMT	32.54	32.60	32.91	34.43	0.00011	0.00008	0.00007	0.00008	852	863	824	865
MAD-MU	35.01	34.50	32.00	36.02	0.00008	0.00008	0.00009	0.00008	880	841	900	788
MAD-RS	41.22	36.00	31.79	33.86	0.00010	0.00008	0.00008	0.00007	886	880	908	890
THR-MC	33.09	33.80	33.99	34.67	0.00011	0.00010	0.00007	0.00008	896	899	881	854
THR-MMT	36.34	35.76	31.96	38.77	0.00011	0.00008	0.00009	0.00008	894	885	853	867
THR-MU	30.89	31.53	32.41	36.43	0.00009	0.00010	0.00008	0.00009	884	870	891	867
THR-RS	37.87	35.00	30.82	38.65	0.00008	0.00008	0.00009	0.00009	850	854	917	932

This study focuses on energy consumption even though discussions are provided for SLA violations and VM migrations. The table 3 and figure 6 (a), (b) and (c) shows that relatively the ECS algorithm shows less energy consumption for MU and MMT. With MC and RS, the performances are more or less comparable. Here the overload of the system is detected using the effective overload detection algorithm. Once the overload is detected, the VM selection is performed using the effective VM selection policies. The VM placement is performed with Lévy flight option. The study uses the values of Status Index (SI) which contains the CPU utilization prior to a VM selection. These strategies will help to produce a better result. In a similar way, there could be more exploration of these contributing policies. Figure 6 (a). Shows the comparison of Energy consumption among ECS, GA, OFS and AC for VM selection policy namely Minimum Utilization (MU)). It gives better result than other algorithms. The lowest energy consumption for ECS is provided by VM selection policies LR-MU

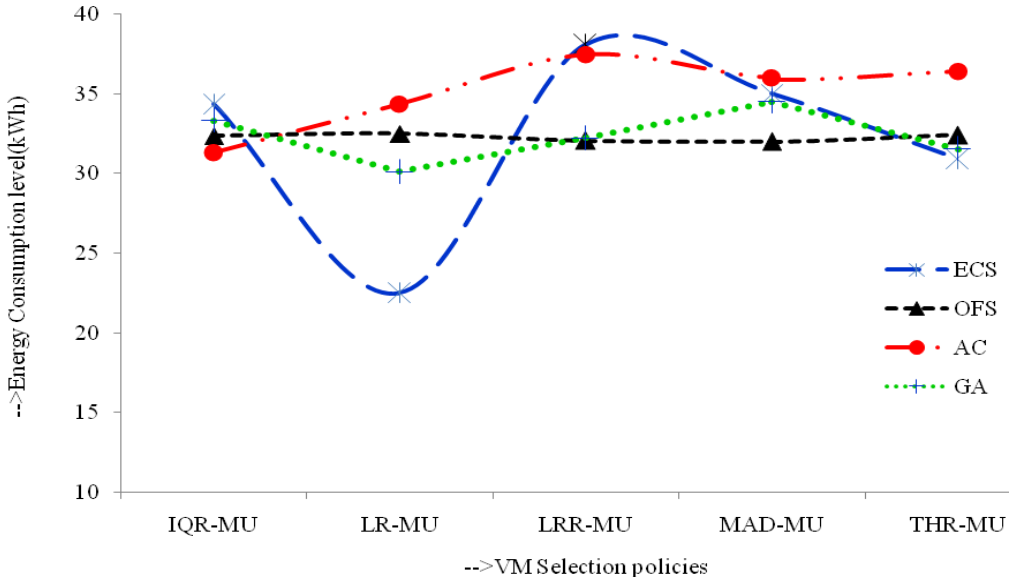

Figure 6 (a). Comparison of Energy consumption by ECS, GA, OFS and AC (for VM selection policy Minimum Utilization (MU))

Figure 6 (b) shows another the comparison of Energy consumption among ECS, GA, OFS and AC under another different promising VM selection policies namely Minimum Migration Time (MMT) and Minimum Utilization (MU). Here too, the proposed ECS give even better result than other algorithms. The lowest energy consumption for ECS is provided by VM selection policies LR-MU.

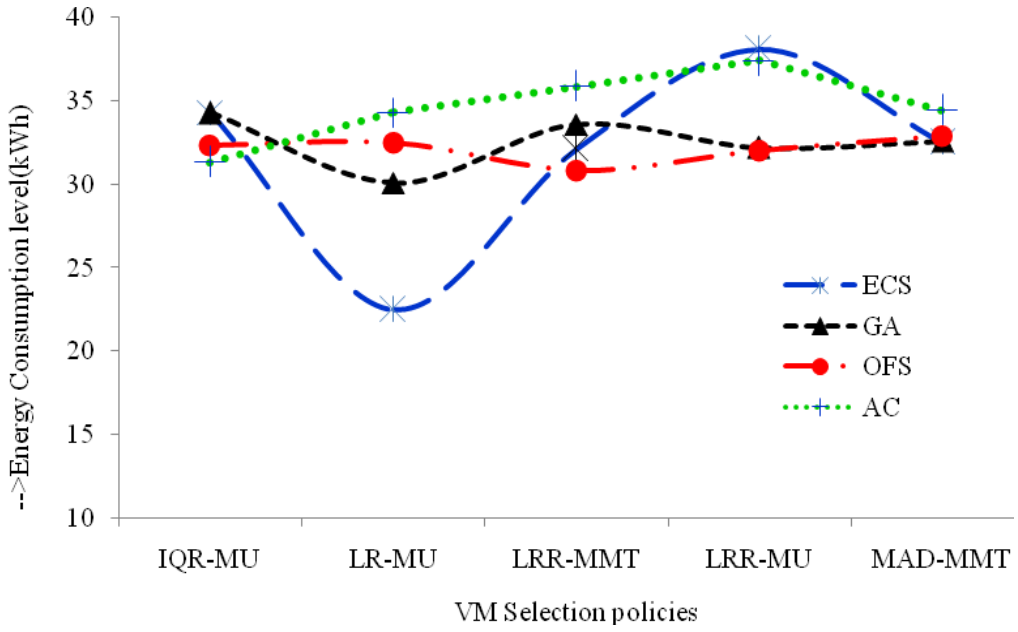

Figure 6 (b). Comparison of Energy consumption by ECS, GA, OFS and AC (for VM selection policies Minimum Migration Time (MMT) and Minimum Utilization (MU)

The figure 6(c) is drawn with logarithmic trend line to show a closer difference in the performances. The proposed algorithm is significantly better than others under VM selection policies IQR-MU, LR-MU and LRR-MU.

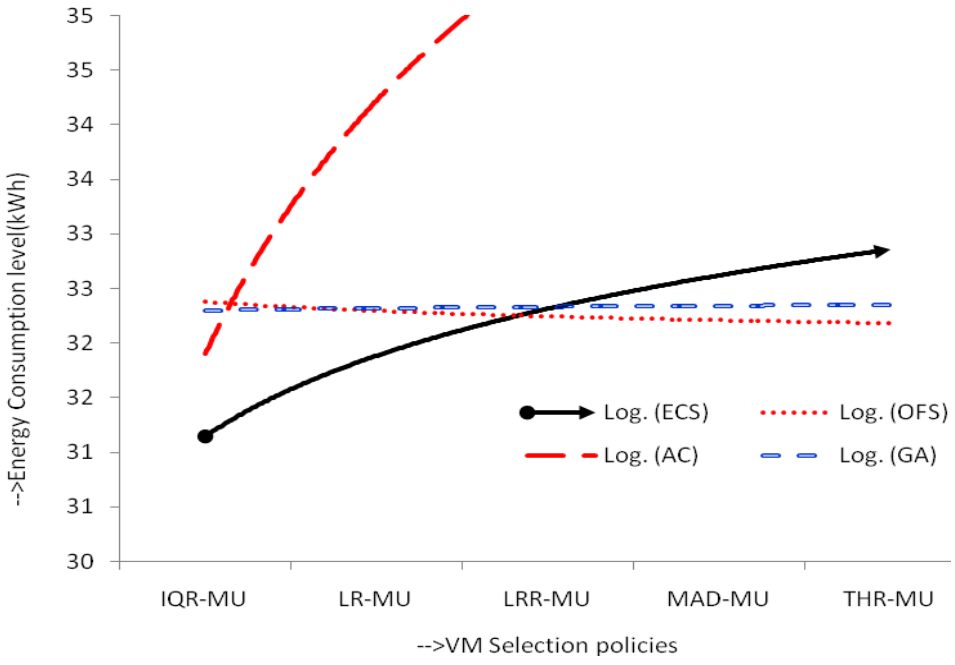

Figure 6 (c). A closed comparison of Energy consumption by ECS, GA, OFS and AC (for VM selection policy Minimum Utilization (MU))

Figure 7 shows a comparison of SLA violation by ECS, GA, OFS and AC. Here the contributing parameters or policies are LR-MMT (AC algorithm), LRR-RS (ECS) etc. Only some few values are plotted as to fit the figures properly.

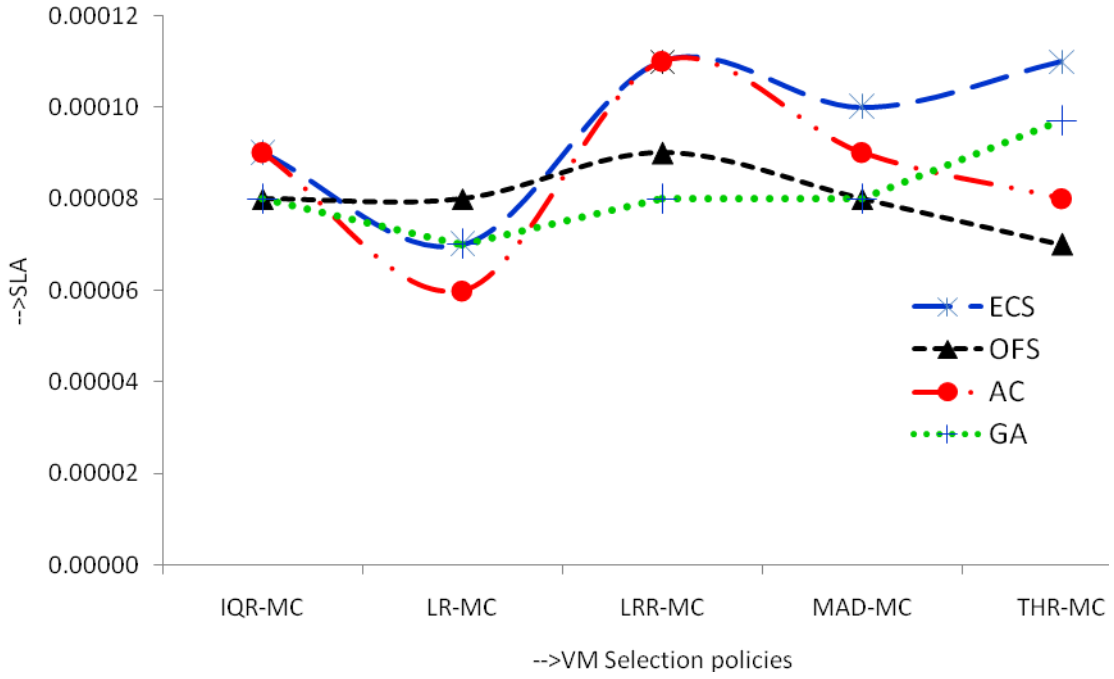

Figure 7. Comparison of SLA Violation by ECS, GA, OFS and AC

Figure 8 shows the numbers of migrations that took place during execution of the different chosen algorithms.

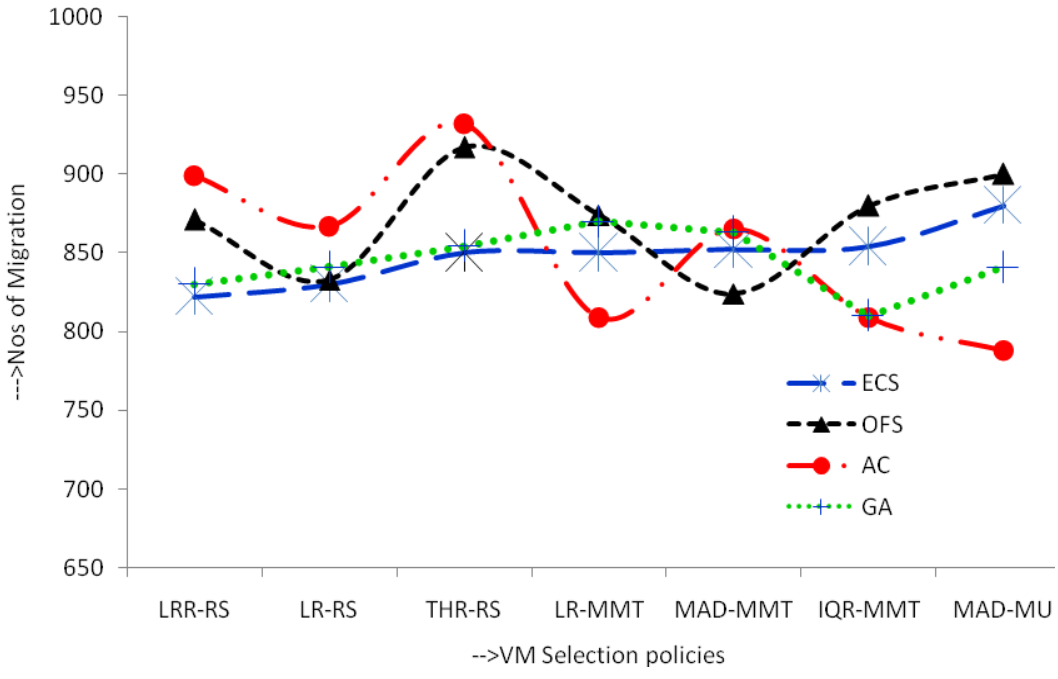

Figure 8. Comparison of VM Migration of ECS, GA, OFS and AC

Some of the contributing parameter or policies used are LR-RS (ECS) and MAD-MU (AC). As shown in the figures and tables, the ECS algorithm with the merits of (1) VM placement with Lévy flight, (2) algorithms for overload detections, (3) policies of VM selection, (4) use of Status Index (SI) for CPU utilization prior to a VM selection, gives better result with energy consumption on VM selection policies like Minimum Migration Time (MMT) and Minimum Utilization (MU). For SLA and VM migration the ECS gave a steady performance graph which is neither too low nor too high.

VI. Conclusion

The VM placement has become an important research problem due to the broadening predominance of big cloud computing data centers and also to maximize the return on investment (ROI) of cloud providers by reducing the overall energy consumption. To competently and efficiently place VMs in the free computing resources, we proposed a new VM placement algorithm called ECS which is inspired by the cuckoo search method. The aim of our proposed work is to reduce the total energy consumption and resource wastage in cloud data center. The efficiency of the proposed ECS algorithm is evaluated through extensive simulations in CloudSim3.0.3 using workload traces from PlanetLab. ECS gave better result with various combinations of VM selection and overload detection policies. This work showed better energy metric for evaluating the performance of the proposed ECS algorithm with VM selection policies like Minimum Migration Time (MMT) and Minimum Utilization (MU), along with steady performance for SLA and VM migration. The results of the comparison confirmed that ECS performs generally well when the numbers of VMs are less (i.e. with less workload). Such algorithm can perform efficiently even in a large scale, dynamic cloud setting because of its capability to utilize Levy flights scheme. This work can be further improved by implementing it in a real heterogeneous environment and also by taking into account some more performance evaluation metrics.

References

[1] Q. Zhang, L. Cheng, R. Boutaba, “Cloud computing: state-of-the-art and research challenges”, J. Internet Services Appl. 1 (1), 718, 2010.

[2] R. Buyya, CS Yeo, S. Venugopal, J. Broberg, I. Brandic, “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility”, Future Generation Computer Systems; 25(6):599–616, 2009

[3] X. Fan, WD Weber, LA Barroso, “Power provisioning for a warehouse-sized computer”, Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA 2007), ACM New York, NY, USA, 2007; 13–23.

[4] M. Randles, D. Lamb, E. Odat, A. Taleb-Bendiab, Distributed redundancy and robustness in complex systems, J. Comput. System Sci. 77 (2), 293–304, 2011

[5] A. Beloglazovy, R.Buyya,Y. C. Lee,A. Zomaya, “A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems“, Technical Report, CLOUDS-TR-2010-3, Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, 2010.

[6] A.Beloglazov, and R.Buyya, "Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers", Concurrency and Computation: Practice and Experience (CCPE), V 24(13), Pp: 1397-1420, John Wiley & Sons, Ltd, 2012

[7] XS Yang, S Deb "Cuckoo search: recent advances & applications." Neural Computing & Application 24.1, 2014

[8] M. Cardosa, M. Korupolu, A. Singh, Shares and utilities based power consolidation in virtualized server environments, in: Proceedings of IFIP/IEEE Integrated Network Management (IM’09), pp. 327–334, 2009

[9] J.Békési, G.Galambos, H.Kellerer, A5/4 linear time bin packing algorithm, J.Comput. System Sci. 60(1) 145–160,2000.

[10] C. Canali; R. Lancellotti, A comparison of techniques to detect similarities in cloud virtual machines, Int. J. of Grid and Utility Computing, 2016 Vol.7, No.2, pp.152 - 162

[11] A. Verma, P. Ahuja, A. Neogi, pMapper: power and migration cost aware application placement in virtualized systems, in: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, pp. 243–264, 2008

[12] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for cloud computing, in Proceedings of HotPower'08 Workshop on Power Aware Computing and Systems, 2008.
[13] B. Li, J. Li, J. Huai, T. Wo, Q. Li, L. Zhong, Enacloud: an energy-saving application live placement approach for cloud computing environments, in: Proceedings of the IEEE International Conference on Cloud Computing, pp. 17–24,2009.
[14] E. Feller, L. Rilling, C. Morin, Energy-aware ant colony based workload placement in clouds, in: Proceedings of the IEEE/ACM International Conference on Grid Computing (GRID), 2011, pp. 26–33
[15] J. Hu, J.Gu, G.Sun, Tianhai Zhao, A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment, IEEE Third International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), pp. 89 – 96,2010
[16] S. Sawant. A Genetic Algorithm Scheduling Approach for Virtual Machine Resources in a Cloud Computing Environment, Master's Projects. Paper 198. http://scholarworks.sjsu.edu/etd_projects/198, 2011
[17] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, L. Yuan, "Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers", in: Proceedings of the IEEE International Conference on Services Computing, pp. 514–521,2010
[18] J. Xu, J. Fortes, "Multi-objective virtual machine placement in virtualized data center environments", in: Proceedings of the IEEE/ACM International Conference on Green Computing and Communications & 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing, p. 179–188, 2010
[19] K.Mukherjee, G.Sahoo(2010), "Green Cloud: An Algorithmic Approach", International Journal of Computer Applications (0975 – 8887), Volume 9– No.9, 2010.
[20] Y. Gao, H. Guan, Z. Qi, Y. Hou, L.Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing", Journal of Computer and System Sciences,Volume 79, Issue 8, Pages:1230–1242, 2013
[21] W. Vogels, Beyond server consolidation, ACM Queue 6 (1) , 20–26, 2008
[22] K. Li, H. Shen, Proxy placement problem for coordinated en-route transcoding proxy caching, Comput. Systems Sci. Engrg. 19 (6) 327–335,2004
[23] K. Li, H. Shen, Optimal proxy placement for coordinated en-route transcoding proxy caching, IEICE Trans. Inform. Syst. 87 (12) 2689–2696, 2004
[24] K. Li, H. Shen, F. Chin, S. Zheng, Optimal methods for coordinated enroute web caching for tree networks, ACM Trans. The internet Technol. (TOIT) 5 (3) 480–507, 2005
[25] K. Li, H. Shen, F. Chin, W. Zhang, Multimedia object placement for transparent data replication, IEEE Trans. Parallel Distrib. Syst. 18 (2) 212–224, 2007
[26] Q. Xilong; X. Peng, An energy-efficient virtual machine scheduler based on CPU share-reclaiming policy, Int. J. of Grid and Utility Computing, 2015 Vol.6, No.2, pp.113 - 120
[27] S. Chaisiri, B. Lee, D. Niyato, Optimal virtual machine placement across multiple cloud providers, in: Proceedings of the IEEE Asia-Pacific Services Computing Conference, pp. 103–110, 2009
[28] B. Speitkamp, M. Bichler, A mathematical programming approach for server consolidation problems in virtualized data centers, IEEE Trans. Services Comput. 266–278, 2010
[29] TM. Lynar, RD. Herbert, S.Chivers, William J. Chivers, Resource allocation to conserve energy in distributed computing, Int. J. of Grid and Utility Computing, 2011 Vol.2, No.1, pp.1 - 10
[30] H. Van, F. Tran, J. Menaud, Performance and power management for cloud infrastructures, in: Proceedings of the IEEE 3rd International Conference on Cloud Computing, , pp. 329–336, 2010
[31] F. Hermenier, X. Lorca, J. Menaud, G. Muller, J. Lawall, Entropy: a consolidation manager for clusters, in: Proc of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, pp. 41–50, 2009
[32] XS. Yang, Engineering Optimisation: An Introduction with Metaheuristic Applications. John Wiley and Sons, 2010
[33] XS. Yang, S. Deb, "Cuckoo search via Lévy flights," in Proceedings of World Congress on Nature and Biologically Inspired Computing (NaBIC 2009), IEEE Publications, USA, pp. 210-214, 2009.
[34] XS. Yang, and S. Deb, "Engineering optimization by the cuckoo search," Int. J. Math. Model. Num. Opt., Vol.1, no. 4, pp. 330–343, 2010.
[35] X.S. Yang, S. Deb, "Multi-objective cuckoo search for design optimization," Comput. Oper. Res., Vol. 40, no. 6, pp.1616–1624, 2013.
[36] I. Pavlyukevich, "Lévy flights, non-local search & simulated annealing," J. Comput. Phys, Vol.226, pp1830, 2007.
[37] XS. Yang, "Cuckoo Search and Firefly Algorithm: Overview and Analysis," in "Cuckoo Search and Firefly Algorithm: Theory and Applications" (Ed. Xin-She Yang),Studies in Computational Intelligence, Vol. 516, Springer International Publishing Switzerland, ,Ch. 1. 2014
[38] P. Civicioglu, and E. Besdok, " A conception comparison of the cuckoo search, particle swarm optimization, differential evolution and artificial bee colony algorithms," Artif. Intell. Rev., 2011. doi:10.1007/s10462-011-92760
[39] P.R. Srivastava, M. Chis, S. Deb, X.S. Yang, " An efficient optimization algorithm for structural software testing," Int. J. Artif. Intell. , Vol. 9, no. S12, pp.68–77, 2012.

[40] S. Walton, O. Hassan, K. Morgan, and M.R. Brown, " Modified cuckoo search: a new gradient free optimization algorithm," Chaos, Solitons Fractals, Vol. 44, no. 9, pp. 710-718, 2011
[41] A. H. Gandomi, X. S. Yang, A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," Engineering with Computers, Vol. 29, no. 1, pp. 17-35, 2013. doi:10.1007/s00366-011-0241-y
[42] A. H. Gandomi, X.S. Yang, S. Talatahari, S. Deb, " Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization," Comput. Math. Appl. Vol. 63, no. 1, pp. 191–200, 2012.
[43] M.F. Shlesinger, G.M. Zaslavsky, and U. Frisch, " Lévy Flights and RelatedTopics in physics,":(Nice, 27–30 June 1994). Springer, New York, 1995.
[44] C.T. Brown, L.S. Liebovitch, R. Glendon, "Lévy flights in dobe ju'hoansi foraging patterns," Hum Ecol, Vol. 35, no. 1, pp. 129–138, 2007.
[45] R.B. Payne, M.D. Sorenson, "The Cuckoos," Oxford University Press, Oxford, Vol. 15, 2005.
[46] A. Ouaarab, B. Ahiod and XS. Yang, "Improved and Discrete Cuckoo Search for Solving the Travelling Salesman Problem," in "Cuckoo Search and Firefly Algorithm: Theory and Applications" Springer, ,2014, 63-84.
[47] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in Clouds: A performance evaluation," in Proceedings of the 1st International Conference on Cloud Computing, pp. 1–12, 2009
[48] XS.Yang, X He,"Firefly algorithm: recent advances and applications." International Journal of Swarm Intelligence 1.1, 36-50, 2013
[49] XS.Yang. "Firefly algorithms for multimodal optimization. In: Stochastic Algorithms: Foundations and Applications, SAGA 2009," Lecture Notes in Computer Sciences, Vol (5792), 169–178, 2009
[50] H. Nakada, T. Hirofuchi, H. Ogawa, and S. Itoh.Toward virtual machine packing optimization based on genetic algorithm. In IWANN'09: Proc. of the 10th International Work-Conference on Artificial Neural Networks, Springer. pp 651-654, 2008.
[51] Y. Gao, H. Guan, Z. Qi, Yang, L Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing", J. of Computer & System Sciences, Vol.79(8), pp.1230–1242,2013.
[52] S. Agrawal, SK Bose, and S.Sundarrajan. Grouping genetic algorithm for solving the serverconsolidation problem with conflicts. In GEC'09: Proc. of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, pp1-88, 2009.
[53] Q.Cao, S. Fujita, Load-balancing schemes for a hierarchical peer-to-peer file search system, Int. J. of Grid and Utility Computing, 2011 Vol.2, No.2, pp.164 - 171
[54] Q Li, Y Hua, W He, D Feng, Z Nie, Y. Sun, "Necklace: An efficient cuckoo hashing scheme for cloud storage services." 2014 IEEE 22nd International Symposium of Quality of Service (IWQoS). IEEE, 2014.

Biographies:

Esha Barlaskar completed her masters degree in Computer Science with specialization in Artificial Intelligence from Assam Don Bosco University, India. Currently Esha is a PhD student in the High Performance and Distributed Computing (HPDC) cluster at Queen's University of Belfast, United Kingdom. Esha's research interests include: cloud data center monitoring, dynamic resource management in cloud and dynamic consolidation of virtual machines.

Yumnam Jayanta Singh is a professor and head of the Department of Computer Science, Engineering and Information Technology, Assam Don Bosco University, India. He earned his PhD in Computer Science and Information Technology. His areas of research interest are Real Time Distributed Database, Cloud Computing, Digital Signal processing, Data warehousing and mining, etc.

Biju Issac is a senior lecturer at the School of Computing, Teesside University, United Kingdom. He earned PhD in Networking and Mobile Communications, along with MCA (Master of Computer Applications) and BE (Electronics and Communications Engineering). His research interests are in computer networks, application of artificial intelligence and optimization problems.